

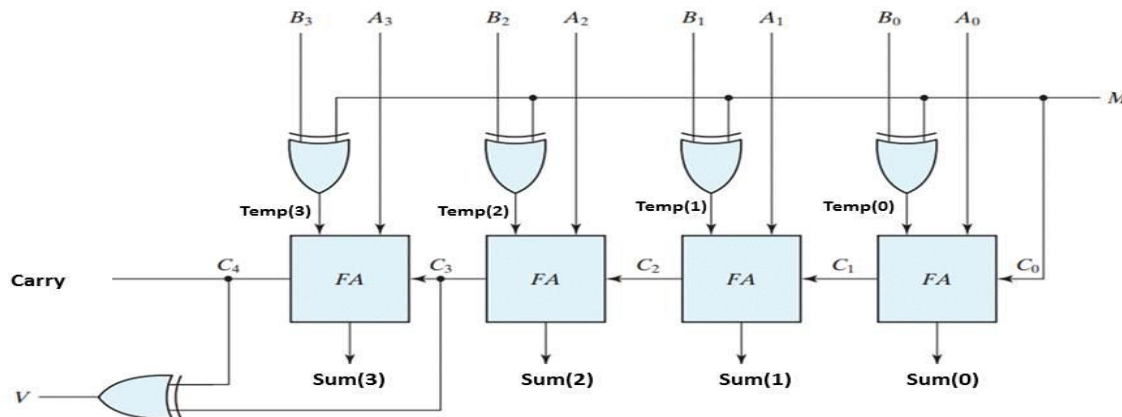
# **VHDL Based Digital Circuits Design**

**Digital Systems Course  
2<sup>nd</sup> Year Students**

**Tanta University  
Faculty of Engineering  
Computer & Control Engineering Department  
2014-2015**

We introduce here some examples of VHDL based digital circuits design for Lab work.

#### 4-Bit Adder/Subtractor



--XOR gate VHDL code

Library ieee;

Use ieee.std\_logic\_1164.all;

Entity xor\_gate is

Port( A, B: in std\_logic;

F: out std\_logic);

End xor\_gate;

Architecture xor\_behav of xor\_gate is

Begin

F <= A xor B;

End xor\_behav;

--Full adder VHDL code

Library ieee;

Use ieee.std\_logic\_1164.all;

Entity FA is

Port( X, Y,Cin: in std\_logic;

Sum,Cout: out std\_logic);

End FA;

Architecture FA\_behav of FA is

Begin

Sum <= (X xor Y) xor Cin;

```
Cout <= (X and Y) or( (X xor Y) and Cin);  
End FA_behav;
```

## **--Top level 4-bit AdderSubtractor**

```
Library ieee;  
Use ieee.std_logic_1164.all;
```

```
Entity Addsub is  
Port( M: in std_logic;  
      A, B: in std_logic_vector(3 downto 0);  
      Sum: out std_logic_vector(3 downto 0);  
      Carry, V: out std_logic);  
End Addsub;
```

```
Architecture Addsub_Sruct of Addsub is
```

```
Component xor_gate is
```

```
Port( A, B: in std_logic;  
      F: out std_logic);
```

```
End component;
```

```
Component FA is
```

```
Port( X, Y,Cin: in std_logic;  
      Sum,Cout: out std_logic);
```

```
End component;
```

```
Signal C :std_logic_vector(4 downto 1);  
Signal temp: std_logic_vector(3 downto 0);
```

```
Begin
```

```
XG1: xor_gate port map (M, B(0),temp(0));  
XG2: xor_gate port map (M, B(1),temp(1));  
XG3:xor_gate port map (M, B(2),temp(2));  
XG4: xor_gate port map (M, B(3),temp(3));
```

```
FA0: FA port map (A(0), temp(0), M, Sum(0),C(1));  
FA1: FA port map (A(1), temp(1), C(1), Sum(1),C(2));  
FA2: FA port map (A(2), temp(2), C(2), Sum(2),C(3));  
FA3: FA port map (A(3), temp(3), C(3), Sum(3),C(4));
```

```
V <= C(3) xor C(4);  
Carry <= C4;
```

```
End Addsub_Struct;
```

## 4-to 1 Mux using If statement

--Neglect enable input

```

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;

ENTITY mux4_1 IS
    PORT (s0           : IN  STD_LOGIC;
          s1           : IN  STD_LOGIC;
          in0          : IN  STD_LOGIC;
          in1          : IN  STD_LOGIC;
          in2          : IN  STD_LOGIC;
          in3          : IN  STD_LOGIC;
          output       : OUT STD_LOGIC
    );
END mux4_1;

ARCHITECTURE if_example OF mux4_1 IS

BEGIN

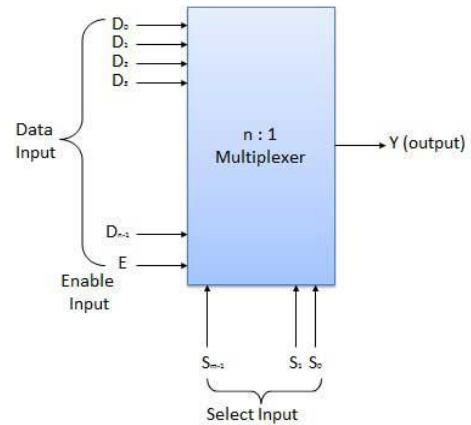
mux:PROCESS(s0, s1, in0, in1, in2, in3)
BEGIN

    IF      (s0='0' AND s1='0') THEN
        output <= in0;
    ELSIF  (s0='1' AND s1='0') THEN
        output <= in1;
    ELSIF  (s0='0' AND s1='1') THEN
        output <= in2;
    ELSIF  (s0='1' AND s1='1') THEN
        output <= in3;
    ELSE      -- (s0 or s1 are not 0 or 1)
        output <= 'X';
    END IF;

END PROCESS mux;

END if_example;

```



## 4-to1 Mux using case statement

```

LIBRARY ieee;
USE ieee.std_logic_1164.ALL;           -- Package declaration.

ENTITY mux4_1 IS
    PORT (s0           : IN  STD_LOGIC;
          s1           : IN  STD_LOGIC;
          in0          : IN  STD_LOGIC;
          in1          : IN  STD_LOGIC;
          in2          : IN  STD_LOGIC;
          in3          : IN  STD_LOGIC;

```

```
        output          : OUT STD_LOGIC
    );
END mux4_1;

ARCHITECTURE case_example OF mux4_1 IS

BEGIN

mux:PROCESS(s0, s1, in0, in1, in2, in3)
    VARIABLE sel : STD_LOGIC_VECTOR(1 DOWNT0 0);
BEGIN
    sel := s1 & s0;    -- concatenate s1 and s0

    CASE sel IS
        WHEN "00" => output <= in0;
        WHEN "01" => output <= in1;
        WHEN "10" => output <= in2;
        WHEN "11" => output <= in3;
        WHEN OTHERS => output <= 'X';
    END CASE;

END PROCESS mux;

END case_example;
```

---

---

## 2-to 1 Mux using (with –select) statement

```
library ieee;
    use ieee.std_logic_1164.all;

entity mux_using_with is
    port (
        din_0    :in  std_logic; -- Mux first input
        din_1    :in  std_logic; -- Mux Second input
        sel      :in  std_logic; -- Select input
        mux_out  :out std_logic  -- Mux output
    );
end entity;

architecture behavior of mux_using_with is

begin
    with (sel) select
        mux_out <= din_0 when '0',
                  din_1 when others;

end architecture;
```

---

---

## 4 to 2 Encoder

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;

entity encod is
Port ( a : in STD_LOGIC_VECTOR (3 downto 0);
      b : out STD_LOGIC_VECTOR (1 downto 0));
end encod;
```

architecture Behavioral of encod is

```
begin
process(a)
begin
if(a(0)='1') then b<="00";
elsif(a(1)='1') then b<="01";
elsif(a(2)='1') then b<="10";
elsif(a(3)='1') then b<="11";
end if;
end process;
end Behavioral;
```

## 16-to 4 Encoder - Using if-else Statement

```
-----
-- Design Name : encoder_using_if
-- File Name   : encoder_using_if.vhd
-----

library ieee;
use ieee.std_logic_1164.all;

entity encoder_using_if is
port (
    enable      :in  std_logic;           -- Enable for the encoder
    encoder_in  :in  std_logic_vector (15 downto 0); -- 16-bit Input
    binary_out  :out std_logic_vector (3 downto 0)  -- 4 bit binary
Output
);
end entity;

architecture behavior of encoder_using_if is

begin
    process (enable, encoder_in) begin
        binary_out <= "0000";
        if (enable = '1') then
            if (encoder_in = X"0002") then binary_out <= "0001"; end if;
            if (encoder_in = X"0004") then binary_out <= "0010"; end if;
            if (encoder_in = X"0008") then binary_out <= "0011"; end if;
            if (encoder_in = X"0010") then binary_out <= "0100"; end if;
```

```
        if (encoder_in = X"0020") then binary_out <= "0101"; end if;
        if (encoder_in = X"0040") then binary_out <= "0110"; end if;
        if (encoder_in = X"0080") then binary_out <= "0111"; end if;
        if (encoder_in = X"0100") then binary_out <= "1000"; end if;
        if (encoder_in = X"0200") then binary_out <= "1001"; end if;
        if (encoder_in = X"0400") then binary_out <= "1010"; end if;
        if (encoder_in = X"0800") then binary_out <= "1011"; end if;
        if (encoder_in = X"1000") then binary_out <= "1100"; end if;
        if (encoder_in = X"2000") then binary_out <= "1101"; end if;
        if (encoder_in = X"4000") then binary_out <= "1110"; end if;
        if (encoder_in = X"8000") then binary_out <= "1111"; end if;
    end if;
end process;
end architecture;
```

---

## 16 to 4 Encoder - Using case Statement

---

```
-- Design Name : encoder_using_case
-- File Name   : encoder_using_case.vhd
-- Function    : Encoder using Case
-----
library ieee;
    use ieee.std_logic_1164.all;

entity encoder_using_case is
    port (
        enable      :in  std_logic;           -- Enable for the encoder
        encoder_in  :in  std_logic_vector (15 downto 0); -- 16-bit Input
        binary_out  :out std_logic_vector (3 downto 0)   -- 4 bit binary
Output
    );
end entity;

architecture behavior of encoder_using_case is
begin
    process (enable, encoder_in) begin
        if (enable = '1') then
            case (encoder_in) is
                when X"0002" => binary_out <= "0001";
                when X"0004" => binary_out <= "0010";
                when X"0008" => binary_out <= "0011";
                when X"0010" => binary_out <= "0100";
                when X"0020" => binary_out <= "0101";
                when X"0040" => binary_out <= "0110";
                when X"0080" => binary_out <= "0111";
                when X"0100" => binary_out <= "1000";
                when X"0200" => binary_out <= "1001";
                when X"0400" => binary_out <= "1010";
                when X"0800" => binary_out <= "1011";
                when X"1000" => binary_out <= "1100";
                when X"2000" => binary_out <= "1101";
                when X"4000" => binary_out <= "1110";
```

```
        when X"8000" => binary_out <= "1111";
        when others  => binary_out <= "0000";
    end case;
end if;
end process;
end architecture;
```

---

## 4 to 2 Priority-Encoder - Using if-else Statement

```
--encoder_in(3) has the highest proirity
library ieee;
    use ieee.std_logic_1164.all;

entity pri_encoder_using_if is
    port (
        enable      :in  std_logic;           -- Enable for the encoder
        encoder_in  :in  std_logic_vector (3 downto 0); -- 4-bit Input
        binary_out  :out std_logic_vector (1 downto 0)  -- 2 bit binary
    );
end entity;

architecture behavior of pri_encoder_using_if is
begin
    process (enable, encoder_in) begin
        binary_out <= "ZZ";
        if (enable = '1') then
            if (encoder_in = 0001") then
                binary_out <= "00";
            elsif (encoder_in = "001X") then
                binary_out <= "01";
            elsif (encoder_in = "01XX") then
                binary_out <= "10";
            elsif (encoder_in = "1XXX") then
                binary_out <= "11";
            else
                binary_out <= "ZZZZ";
            end if;
        end if;
    end process;
end architecture;
```

---



## 16-to 4 Priority-Encoder - Using if-else Statement

```
-----
-- Design Name : pri_encoder_using_if
--encoder_in(0) has the highest proirity
-----

library ieee;
    use ieee.std_logic_1164.all;

entity pri_encoder_using_if is
    port (
        enable      :in  std_logic;           -- Enable for the encoder
        encoder_in  :in  std_logic_vector (15 downto 0); -- 16-bit Input
        binary_out  :out std_logic_vector (3 downto 0)  -- 4 bit binary
Output
    );
end entity;

architecture behavior of pri_encoder_using_if is

begin
    process (enable, encoder_in) begin
        binary_out <= "0000";
        if (enable = '1') then
            if (encoder_in = "XXXXXXXXXXXXXXXXX1") then
                binary_out <= "0000";
            if (encoder_in = "XXXXXXXXXXXXXXXXX10") then
                binary_out <= "0001";
            elsif (encoder_in = "XXXXXXXXXXXXXXXXX100") then
                binary_out <= "0010";
            elsif (encoder_in = "XXXXXXXXXXXXXXXXX1000") then
                binary_out <= "0011";
            elsif (encoder_in = "XXXXXXXXXXXXXXXXX10000") then
                binary_out <= "0100";
            elsif (encoder_in = "XXXXXXXXXXXXX100000") then
                binary_out <= "0101";
            elsif (encoder_in = "XXXXXXXXXXXXX1000000") then
                binary_out <= "0110";
            elsif (encoder_in = "XXXXXXXXXXXXX10000000") then
                binary_out <= "0111";
            elsif (encoder_in = "XXXXXXX100000000") then
                binary_out <= "1000";
            elsif (encoder_in = "XXXXXXX1000000000") then
                binary_out <= "1001";
            elsif (encoder_in = "XXXXXX10000000000") then
                binary_out <= "1010";
            elsif (encoder_in = "XXXXX100000000000") then
                binary_out <= "1011";
            elsif (encoder_in = "XXX1000000000000") then
                binary_out <= "1100";
            elsif (encoder_in = "XX10000000000000") then
                binary_out <= "1101";
            elsif (encoder_in = "X100000000000000") then
                binary_out <= "1110";
            elsif (encoder_in = "1000000000000000") then
                binary_out <= "1111";
        end if;
    end if;
end process;
```

```
        else
            binary_out <= "ZZZZ";
        end if;
    end if;
end process;
end architecture;
```

---

## 16to 4 Priority Encoder - Using when Statement

---

```
-- Design Name : pri_encoder_using_when
-- encoder_in(0) has the highest proirity
-----
library ieee;
    use ieee.std_logic_1164.all;

entity pri_encoder_using_when is
    port (
        enable      :in  std_logic;           -- Enable for the encoder
        encoder_in   :in  std_logic_vector (15 downto 0); -- 16-bit Input
        binary_out   :out std_logic_vector (3 downto 0)   -- 4 bit binary
        Output
    );
end entity;

architecture behavior of pri_encoder_using_when is

begin
    binary_out <= "0000" when (enable = '0') else
        "0000" when (encoder_in = "XXXXXXXXXXXXXXXX1") else
        "0001" when (encoder_in = "XXXXXXXXXXXXXXXX10") else
        "0010" when (encoder_in = "XXXXXXXXXXXXXXXX100") else
        "0011" when (encoder_in = "XXXXXXXXXXXXXXXX1000") else
        "0100" when (encoder_in = "XXXXXXXXXXXXXXXX10000") else
        "0101" when (encoder_in = "XXXXXXXXXXXXXXXX100000") else
        "0110" when (encoder_in = "XXXXXXXXXX10000000") else
        "0111" when (encoder_in = "XXXXXXXXXX10000000") else
        "1000" when (encoder_in = "XXXXXXX1000000000") else
        "1001" when (encoder_in = "XXXXXXX1000000000") else
        "1010" when (encoder_in = "XXXXXX100000000000") else
        "1011" when (encoder_in = "XXXXX1000000000000") else
        "1100" when (encoder_in = "XXX10000000000000") else
        "1101" when (encoder_in = "XX100000000000000") else
        "1110" when (encoder_in = "X1000000000000000") else
        "1111";

end architecture;
```

---

## 2 to 4 Decoder

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
```

```
entity decode_2to4_top is
    Port ( A : in STD_LOGIC_VECTOR (1 downto 0); -- 2-bit input
          X : out STD_LOGIC_VECTOR (3 downto 0); -- 4-bit output
          EN : in STD_LOGIC); -- enable input
end decode_2to4_top;

architecture Behavioral of decode_2to4_top is
begin
    process (A, EN)
    begin
        X <= "0000"; -- default output value
        if (EN = '1') then -- active high enable pin
            case A is
                when "00" => X(0) <= '1';
                when "01" => X(1) <= '1';
                when "10" => X(2) <= '1';
                when "11" => X(3) <= '1';
                when others => X <= "0000";
            end case;
        end if;
    end process;
end Behavioral;
```

---

## 4 to 16 Decoder - Using case Statement

```
library ieee;
use ieee.std_logic_1164.all;

entity decoder_using_case is
    port (
        enable      :in  std_logic; -- Enable for the decoder
        binary_in   :in  std_logic_vector (3 downto 0); -- 4-bit Input
        decoder_out  :out std_logic_vector (15 downto 0) -- 16-bit Output
    );
end entity;

architecture behavior of decoder_using_case is
begin
    process (enable, binary_in) begin
        decoder_out <= X"0000";
        if (enable = '1') then
            case (binary_in) is
                when X"0"   => decoder_out <= X"0001";
                when X"1"   => decoder_out <= X"0002";
                when X"2"   => decoder_out <= X"0004";
                when X"3"   => decoder_out <= X"0008";
                when X"4"   => decoder_out <= X"0010";
                when X"5"   => decoder_out <= X"0020";
                when X"6"   => decoder_out <= X"0040";
                when X"7"   => decoder_out <= X"0080";
                when X"8"   => decoder_out <= X"0100";
                when X"9"   => decoder_out <= X"0200";
                when X"A"   => decoder_out <= X"0400";
```

```
        when X"B"    => decoder_out <= X"0800";
        when X"C"    => decoder_out <= X"1000";
        when X"D"    => decoder_out <= X"2000";
        when X"E"    => decoder_out <= X"4000";
        when X"F"    => decoder_out <= X"8000";
        when others => decoder_out <= X"0000";
    end case;
end if;
end process;
end architecture;
```

---

## 4 to 16 Decoder using (with-select) statement

```
-----
-- Design Name : decoder_using_with
-- File Name    : decoder_using_with.vhd
-- Function     : decoder using with-select
-----

library ieee;
    use ieee.std_logic_1164.all;

entity decoder_using_select is
    port (
        enable      :in  std_logic;           -- Enable for the decoder
        binary_in   :in  std_logic_vector (3 downto 0); -- 4-bit input
        decoder_out  :out std_logic_vector (15 downto 0) -- 16-bit output
    );
end entity;

architecture behavior of decoder_using_select is

begin
    with (binary_in) select
        decoder_out <= X"0001" when X"0",
                        X"0002" when X"1",
                        X"0004" when X"2",
                        X"0008" when X"3",
                        X"0010" when X"4",
                        X"0020" when X"5",
                        X"0040" when X"6",
                        X"0080" when X"7",
                        X"0100" when X"8",
                        X"0200" when X"9",
                        X"0400" when X"A",
                        X"0800" when X"B",
                        X"1000" when X"C",
                        X"2000" when X"D",
                        X"4000" when X"E",
                        X"8000" when X"F",
                        X"0000" when others;

end architecture;
```

A Flip-flop is the basic element which is used to store information of one bit. Flip-flops have their content change either at the rising or falling edge of the enable signal(usually the controlling clock signal).

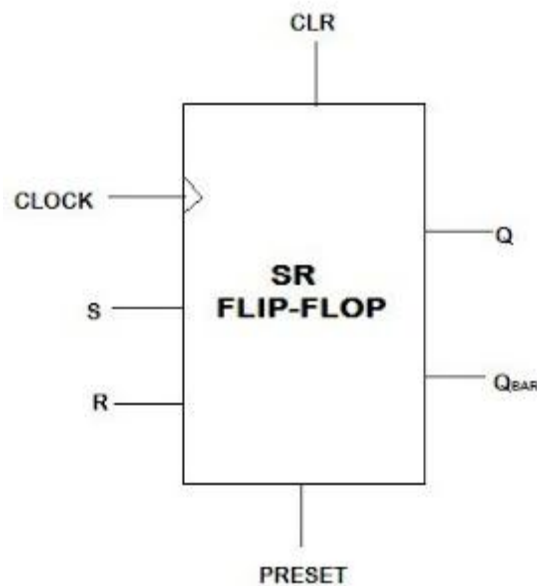
There are basically four main types of flip-flops:

1. SR Flip-flop
2. D Flip-flop
3. JK Flip-flop
4. T Flip-flop.

### 1. SR FLIP-FLOP VHDL Code:

A SR flip flop used in digital electronics will provide the results in a similar manner to the JK flip flop and this is the reason why the vhdl codes for these two flipflops are similar in nature.

Given below is a behavioral approach of writing the code for a SR Flip-flop.



```
library ieee;
use ieee. std_logic_1164.all;
use ieee. std_logic_arith.all;
use ieee. std_logic_unsigned.all;
```

```
entity SR-FF is
PORT( S,R,CLOCK,CLR,PRESET: in std_logic;
      Q, QBAR: out std_logic);
end SR-FF;
```

```
Architecture behavioral of SR-FF is
begin
P1: PROCESS(CLOCK,CLR,PRESET)
variable x: std_logic;
begin
if(CLR='0') then
x:='0';

elsif(PRESET='0')then
x:='1';

elsif(CLOCK='1' and CLOCK'EVENT) then

if(S='0' and R='0')then
x:=x;
elsif(S='1' and R='1')then
x:='Z';

elsif(S='0' and R='1')then
x:='0';

else
x:='1';

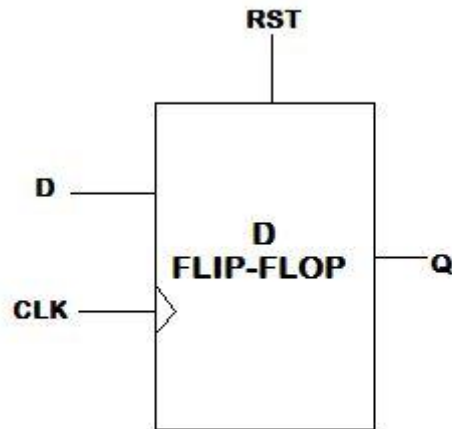
end if;
end if;

Q<=x;
QBAR<=not x;
end PROCESS;
end behavioral;
```

## **2. D FLIP-FLOP VHDL Code:**

A D flip flop or Delay flip flop gives the same output as the input provided and thus the vhdl code is much simpler.

Given below is a behavioral approach of writing the vhdl code for a D Flip-flop.



```
library ieee;
use ieee. std_logic_1164.all;
use ieee. std_logic_arith.all;
use ieee. std_logic_unsigned.all;

entity D-FF is
PORT( D,CLK,RST: in std_logic;
      Q: out std_logic);
end D-FF;

architecture behavioral of D-FF is

begin
P1: process(RST,CLK)

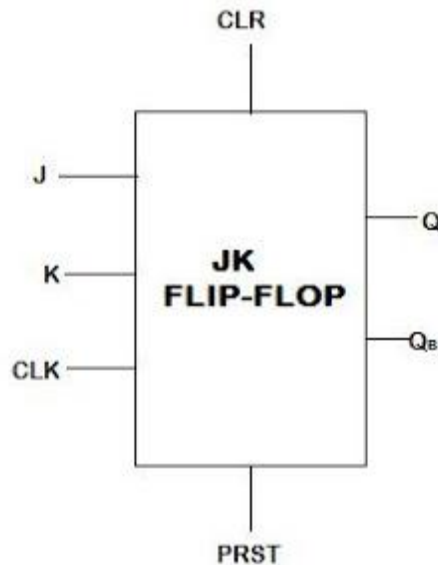
begin

if(RST='1')then
Q<='0';

elseif(CLK='1' and CLK'EVENT) then
Q<=D;
end if; end process;
end behavioral;
```

### 3. JK FLIP-FLOP VHDL Code:

Given below is a behavioral approach of writing the code for a JK Flip-flop.



```
library ieee;
use ieee. std_logic_1164.all;
use ieee. std_logic_arith.all;
use ieee. std_logic_unsigned.all;

entity JK-FF is
PORT( J,K,CLK,PRST,CLR: in std_logic;
      Q, QB: out std_logic);
end JK-FF;
```

```
Architecture behavioral of JK-FF is
begin
P1: PROCESS(CLK,CLR,PRST)
variable x: std_logic;
begin
if(CLR='0') then
x:='0';

elsif(PRST='0')then
```

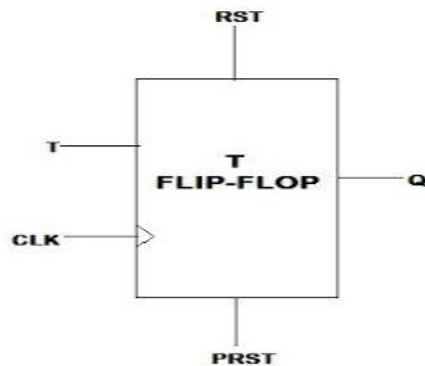


```
x:='1';  
  
elsif(CLK='1' and CLK'EVENT) then  
  if(J='0' and K='0')then  
    x:=x;  
  elsif(J='1' and K='1')then  
    x:= not x;  
  
  elsif(J='0' and K='1')then  
    x:='0';  
  else  
    x:='1';  
  
  end if;  
end if;  
Q<=x;  
QB<=not x;  
end PROCESS;  
end behavioral;
```

#### 4. T FLIP-FLOP VHDL Code:

The T in a t flip flop stands for toggle and this is exactly what this digital component does. It simply toggles the value of a particular input. A basic not gate will solve the problem in the vhd code for this element.

Given below is a behavioral approach of writing the code for a T Flip-flop.



```
library ieee;
use ieee. std_logic_1164.all;
use ieee. std_logic_arith.all;
use ieee. std_logic_unsigned.all;

entity T-FF is

PORT( T,CLK,PRST,RST: in std_logic;
      Q: out std_logic);

end T-FF;

architecture behavioral of T-FF is

begin
P1: process(CLK,PRST,RST)

variable x: std_logic;

begin

if(RST='0') then

x:='0';

elsif(RST='1' and PRST='0') then

x:='1';

elsif(CLK='1' and CLK'EVENT) then

if(T='1')then

x:= not x;

end if;
end if;

Q<=x;

end process;
end behavioral;
```

---

---

### Regular D latch(register)

```
library ieee;
use ieee.std_logic_1164.all;

entity dlatch_reset is
    port (
        data  :in  std_logic; -- Data input
        en    :in  std_logic; -- Enable input
        reset :in  std_logic; -- Reset input
        q     :out std_logic  -- Q output
    );
end entity;

architecture rtl of dlatch_reset is
begin
    process (en, reset, data) begin
        if (reset = '0') then
            q <= '0';
        elsif (en = '1') then
            q <= data;
        else
            null;
        end if;
    end process;
end architecture;
```

---

### 8-bit Parallel to Serial converter

```
-- Description
-- Implements a simple 8-bit parallel to serial converter in VHDL.

library ieee;
use ieee.std_logic_1164.all;

entity PAR2SER is
    port (DIN : in std_logic_vector (7 downto 0); -- input register
          MODE : in std_logic_vector (1 downto 0); -- mode selection
          CLK, RESET : in std_logic; -- clock and reset
          SDOUT : out std_logic); -- output data
end PAR2SER;

-- purpose: Implement main architecture of PAR2SER

architecture BEHAVIOR of PAR2SER is

    signal IDATA : std_logic_vector(7 downto 0); -- internal data
```

```
begin -- BEHAVIOR

-- purpose: Main process
process (CLK, RESET)
begin -- process
-- activities triggered by asynchronous reset (active high)

if RESET = '1' then
SDOUT <= '0';
IDATA <= "00000000";
-- activities triggered by rising edge of clock
elsif CLK'event and CLK = '1' then

case MODE is
when "00" => -- no operation
null;
when "01" => -- load operation
IDATA <= DIN;
when "10" => -- shift left
SDOUT <= IDATA(7);
for mloop in 6 downto 0 loop
IDATA(mloop+1) <= IDATA(mloop);
end loop; -- mloop
when others => -- no operation otherwise
null;
end case;

end if;

end process;

end BEHAVIOR;
```

---

### **BCD to 7-Seg Decoder**

```
library IEEE;

use IEEE.std_logic_1164.all;

use IEEE.std_logic_unsigned.all;

entity DISPLAY_DECODER is

port ( VALUE : in bit_vector(3 downto 0); -- Bit 3 is MSB

ZERO_BLANK : in bit;

DISPLAY : out bit_vector(6 downto 0); -- 7 bit signal

ZERO_BLANK_OUT : out bit);
```

```
end DISPLAY_DECODER;
```

architecture BEHAVIOUR of DISPLAY\_DECODER is

```
begin
```

```
process (VALUE, ZERO_BLANK)    -- sensitivity list
```

```
begin
```

```
case VALUE is -- case-when statement described how decode is
```

```
-- driven based on the value of the input.
```

```
when "0000" => if ZERO_BLANK='1' then
```

```
    DISPLAY <= "0000000";
```

```
    ZERO_BLANK_OUT <= '1';
```

```
else
```

```
    DISPLAY <= "1111110";
```

```
end if;
```

```
when "0001" => DISPLAY <= "0110000";
```

```
when "0010" => DISPLAY <= "1101101";
```

```
when "0011" => DISPLAY <= "1111001";
```

```
when "0100" => DISPLAY <= "0110011";
```

```
when "0101" => DISPLAY <= "1011011";
```

```
when "0110" => DISPLAY <= "1011111";
```

```
when "0111" => DISPLAY <= "1110000";
```

```
when "1000" => DISPLAY <= "1111111";
```

```
when OTHERS => DISPLAY <= "1001111"; -- when others, an error is specified
```

```
end case;
```

```
end process;
```

```
end BEHAVIOUR;
```

## Test bench

```
library IEEE;
```

```
use IEEE.std_logic_1164.all;
```

```
use IEEE.std_logic_unsigned.all;
```

```
entity DISPLAY_DECODER_TB is
```

```
end DISPLAY_DECODER_TB;
```

```
architecture ARC_DISPLAY_DECODER_TB of DISPLAY_DECODER_TB is
```

```
signal VALUE      : bit_vector(3 downto 0);
```

```
signal ZERO_BLANK  : bit;
```

```
signal DISPLAY     : bit_vector(6 downto 0);
```

```
signal ZERO_BLANK_OUT : bit;
```

```
component DISPLAY_DECODER
```

```
port ( VALUE      : in bit_vector(3 downto 0);
```

```
ZERO_BLANK      : in bit;
```

```
DISPLAY         : out bit_vector(6 downto 0);
```

```
ZERO_BLANK_OUT  : out bit);
```

```
end component;
```

```
begin
```

```
INPUT_VALUES: process
```

```
begin
```

```
ZERO_BLANK <= '1';
```

```
VALUE <= "0000";
```

```
wait for 5 ns;
```

```
ZERO_BLANK    <= '0';

VALUE         <= "0000";

wait for 7 ns;

ZERO_BLANK <= '1';

VALUE <= "0010";

wait for 12 ns;

ZERO_BLANK    <= '0';

VALUE <= "0100";

wait for 12 ns;

ZERO_BLANK <= '0';

VALUE <= "0110";

wait for 7 ns;

end process INPUT_VALUES;

U1: DISPLAY_DECODER

port map(VALUE, ZERO_BLANK, DISPLAY, ZERO_BLANK_OUT);

end ARC_DISPLAY_DECODER_TB;

configuration CFG_DISPLAY_DECODER of DISPLAY_DECODER_TB is

for ARC_DISPLAY_DECODER_TB

for U1:DISPLAY_DECODER use entity

work.DISPLAY_DECODER(BEHAVIOUR);

end for;

end for;

end CFG_DISPLAY_DECODER;
```

## A comparator circuit

```
library ieee;
use ieee.std_logic_1164.all;

-----

entity Comparator is

port(   A:      in std_logic_vector(2 downto 0);
        B:      in std_logic_vector(2 downto 0);
        less:    out std_logic;
        equal:   out std_logic;
        greater: out std_logic
);
end Comparator;

architecture behv of Comparator is

begin

    process (A,B)
    begin
        if (A<B) then
            less <= '1';
            equal <= '0';
            greater <= '0';
        elsif (A=B) then
            less <= '0';
            equal <= '1';
            greater <= '0';
        else
            less <= '0';
            equal <= '0';
            greater <= '1';
        end if;
    end process;

end behv;
```

---

## Registers

```
library ieee ;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

-----

entity reg is

port(   I:      in std_logic_vector(1 downto 0);
        clock:  in std_logic;
        load:   in std_logic;
```



```
        clear: in std_logic;
        Q:      out std_logic_vector(1 downto 0)
    );
end reg;

architecture behv of reg is

    signal Q_tmp: std_logic_vector(1 downto 0);

begin

    process(I, clock, load, clear)
    begin

        if clear = '0' then
            -- use 'range in signal assignment
            Q_tmp <= "00";
        elsif (clock='1' and clock'event) then
            if load = '1' then
                Q_tmp <= I;
            end if;
        end if;

    end process;

    -- concurrent statement
    Q <= Q_tmp;

end behv;
```

---

## Shift registers

```
-- 3-bit Shift-Register/Shifter
-- reset is ignored in this code

library ieee ;
use ieee.std_logic_1164.all;

entity shift_reg is
port(   I:          in std_logic;
        clock:      in std_logic;
        shift:      in std_logic;
        Q:          out std_logic
    );
end shift_reg;

architecture behv of shift_reg is

    -- initialize the declared signal
    signal S: std_logic_vector(2 downto 0) := "111";

begin
```

```
process(I, clock, shift, S)
begin

    -- everything happens upon the clock changing
    if clock'event and clock='1' then
        if shift = '1' then
            S <= I & S(2 downto 1);
        end if;
    end if;

end process;

-- concurrent assignment
Q <= S(0);

end behv;
```

---

n-bit counter

-- VHDL code for n-bit counter

```
library ieee ;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
```

entity counter is

```
generic(n: natural :=2);
port(   clock: in std_logic;
       clear: in std_logic;
       count: in std_logic;
       Q:      out std_logic_vector(n-1 downto 0)
);
end counter;
```

architecture behv of counter is

```
    signal Pre_Q: std_logic_vector(n-1 downto 0);

begin

    -- behavior describe the counter

    process(clock, count, clear)
    begin
        if clear = '1' then
            Pre_Q <= Pre_Q - Pre_Q;
        elsif (clock='1' and clock'event) then
            if count = '1' then
                Pre_Q <= Pre_Q + 1;
            end if;
        end if;
    end process;
```

```
-- concurrent assignment statement  
Q <= Pre_Q;  
  
end behv;
```

---